ASSIGNMENT NO.3

Note:

- The assignment is designed to practice class, fields, and methods only.
- Create a separate project for each question.
- Do not use getter/setter methods or constructors for these assignments.
- Define two classes: one class to implement the logic and another class to test it.

1. Loan Amortization Calculator

Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:

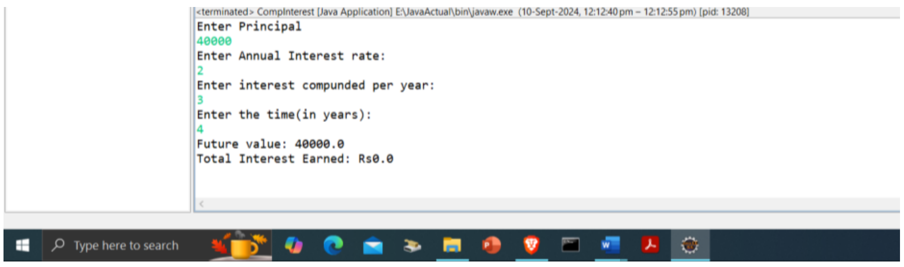
1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
   - **Monthly Payment Calculation:**
     - `monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((1 + monthlyInterestRate)^(numberOfMonths) - 1)`
     - Where `monthlyInterestRate = annualInterestRate / 12 / 100` and `numberOfMonths = loanTerm * 12`
     - Note: Here ^ means power and to find it you can use Math.pow( ) method
3. Display the monthly payment and the total amount paid over the life of the loan, in Indian Rupees (₹).

Define class LoanAmortizationCalculator with methods acceptRecord, calculateMonthlyPayment & printRecord and test the functionality in main method.

Ans:

```
class LACalci{
//	1.	Accept the principal amount (loan amount), annual
interest rate, and loan term (in years) from the user
	private float pa;
	private float rate;
	private float time;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter Principle Amount: ");
		this.pa=sc.nextFloat();
		System.out.println("Enter Interest rate: ");
		this.rate=sc.nextFloat();
		System.out.println("Enter time (in years): ");
		this.time=sc.nextFloat();

	}

	public void calculateMonthlyPayment () {
//		o	Monthly Payment Calculation:
```

```
//                ♣    monthlyPayment = principal * 
(monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / 
((1 + monthlyInterestRate)^(numberOfMonths) - 1)
//                ♣    Where monthlyInterestRate = annualInterestRate 
/ 12 / 100
//                   and numberOfMonths = loanTerm * 12
//                ♣    Note: Here ^ means power and to find it you 
can use Math.pow( ) method

            float montlyPayment= (float) ((float)
(pa*(rate*Math.pow((1+rate), (time))))/((1+Math.pow(rate, time)-
1)));
            float monthlyInterestRate=rate/12/100;
            float noMonth=rate*12;

            display(montlyPayment,monthlyInterestRate,noMonth);
      }

      public void display(float mp,float mi, float nm) {
//          displaying whatever we calculated
            System.out.println("Your Monthly payments: "+mp);
            System.out.println("Your Monthly interest rate: "+mi);
            System.out.println("Your No. Month: "+nm);
      }
}

public class Assignment2 {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
            LACalci l1=new LACalci();
            l1.acceptRecord();
            l1.calculateMonthlyPayment();


      }

}
```



```
<terminated> CompInterest [Java Application] E:\JavaActual\bin\javaw.exe (10-Sept-2024, 12:12:40 pm – 12:12:55 pm) [pid: 13208]
Enter Principal
40000
Enter Annual Interest rate:
2
Enter interest compunded per year:
3
Enter the time(in years):
4
Future value: 40000.0
Total Interest Earned: Rs0.0
```

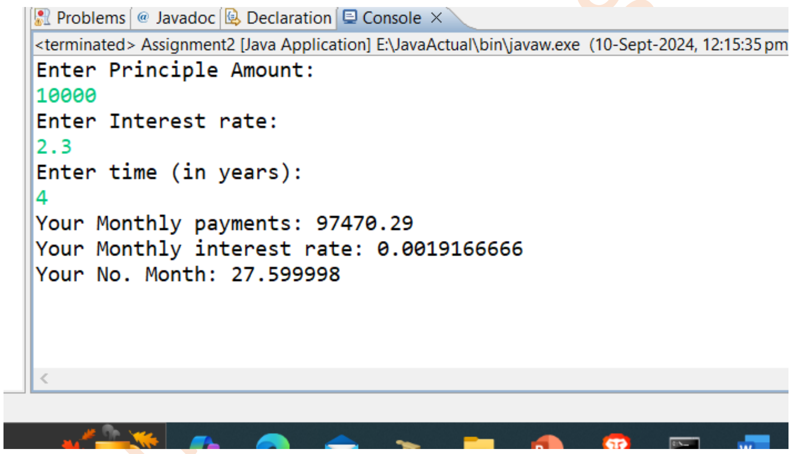
2. Compound Interest Calculator for Investment

Develop a system to compute the future value of an investment with compound interest. The system should:

1. Accept the initial investment amount, annual interest rate, number of times the interest is compounded per year, and investment duration (in years) from the user.
2. Calculate the future value of the investment using the formula:
   - **Future Value Calculation:**
     - `futureValue = principal * (1 + annualInterestRate / numberOfCompounds)^(numberOfCompounds * years)`
   - **Total Interest Earned:** `totalInterest = futureValue - principal`
3. Display the future value and the total interest earned, in Indian Rupees (₹).

Define class CompoundInterestCalculator with methods acceptRecord , calculateFutureValue, printRecord and test the functionality in main method.

Ans:
```
public class CompInterest {
        private float ia;
        private float air;
        private float nc;
        private float time;

        void acceptRecord() {
//
                Scanner sc = new Scanner(System.in);
                System.out.println("Enter Principal");
                this.ia = sc.nextFloat();
                System.out.println("Enter Annual Interest rate: ");
                this.air = sc.nextFloat();
                System.out.println("Enter interest compunded per year:");
                this.nc = sc.nextFloat();
                System.out.println("Enter the time(in years): ");
                this.nc = sc.nextFloat();
                sc.close();


        }

        void FutureValue() {
//              2.      Calculate the future value of the investment using the formula:
                //o     Future Value Calculation:
                //♣     futureValue = principal * (1 + annualInterestRate /
numberOfCompounds)^(numberOfCompounds * years)
                //o     Total Interest Earned: totalInterest = futureValue - principal
                float future=(float) (ia *Math.pow((1+air/nc), (nc*time) ));
                float totalintr=future-ia;
                diplay(future,totalintr);
        }

        private void diplay(float future, float totalintr) {
                //3.      Display the future value and the total interest earned, in Indian Rupees
(₹).
                //Define class CompoundInterestCalculator with methods acceptRecord ,
calculateFutureValue, printRecord and test the functionality in main method.

                System.out.println("Future value: "+future);
                System.out.println("Total Interest Earned: Rs"+totalintr);

        }

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                CompInterest c=new CompInterest();
                c.acceptRecord();
                c.FutureValue();
        }

}
```


Problems | Javadoc | Declaration | Console

```
<terminated> Assignment2 [Java Application] E:\JavaActual\bin\javaw.exe (10-Sept-2024, 12:15:35 pm
Enter Principle Amount:
10000
Enter Interest rate:
2.3
Enter time (in years):
4
Your Monthly payments: 97470.29
Your Monthly interest rate: 0.0019166666
Your No. Month: 27.599998
```

3. BMI (Body Mass Index) Tracker

Create a system to calculate and classify Body Mass Index (BMI). The system should:

1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5

   - Normal weight: 18.5 ≤ BMI < 24.9
   - Overweight: 25 ≤ BMI < 29.9
   - Obese: BMI ≥ 30
4. Display the BMI value and its classification.

Define class BMITracker with methods acceptRecord, calculateBMI, classifyBMI & printRecord and test the functionality in main method.

4. Discount Calculation for Retail Sales

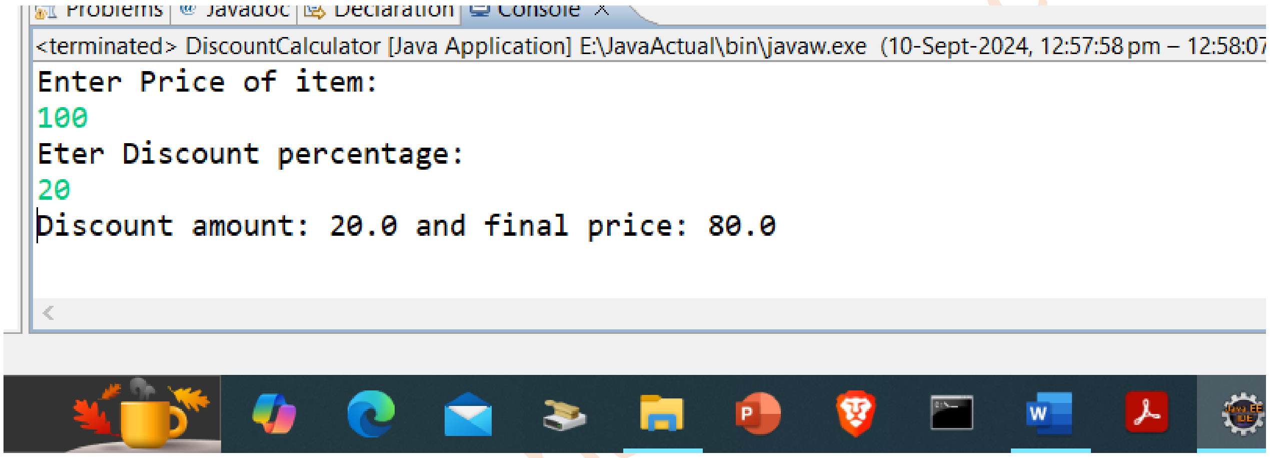
Design a system to calculate the final price of an item after applying a discount. The system should:

1. Accept the original price of an item and the discount percentage from the user.
2. Calculate the discount amount and the final price using the following formulas:
   - **Discount Amount Calculation:** `discountAmount = originalPrice * (discountRate / 100)`
   - **Final Price Calculation:** `finalPrice = originalPrice - discountAmount`
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

Define class DiscountCalculator with methods acceptRecord, calculateDiscount & printRecord and test the functionality in main method.

Ans:
```
public class DiscountCalculator {
	float price;
	float dp;
//	1.	Accept the original price of an item and the discount percentage from the user.
	void accpet() {
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter Price of item: ");
		this.price=sc.nextFloat();
		System.out.println("Eter Discount percentage: ");
		this.dp=sc.nextFloat();
		sc.close();
		discount_logic(price,dp);
	}
private void discount_logic(float price2, float dp2) {
//	2.	Calculate the discount amount and the final price using the following formulas:
//	o	Discount Amount Calculation: discountAmount = originalPrice * (discountRate / 100)
//	o	Final Price Calculation: finalPrice = originalPrice - discountAmount
	float da=price2*(dp2/100);
	final float fprice=price2-da;
	display(da,fprice);

}
```

```
        private void display(float da, float fprice) {
//              3.      Display the discount amount and the final price of the item, in Indian
Rupees (₹).
                System.out.println("Discount amount: "+ da +" and final price: "+ fprice);
}
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                DiscountCalculator d= new DiscountCalculator();
                d.accpet();
        }

}
```



```
<terminated> DiscountCalculator [Java Application] E:\JavaActual\bin\javaw.exe  (10-Sept-2024, 12:57:58 pm – 12:58:07
Enter Price of item:
100
Eter Discount percentage:
20
Discount amount: 20.0 and final price: 80.0
```

5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:

1. Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
2. Accept the number of vehicles of each type passing through the toll booth.
3. Calculate the total revenue based on the toll rates and number of vehicles.
4. Display the total number of vehicles and the total revenue collected, in Indian Rupees (₹).

- **Toll Rate Examples:**
  - Car: ₹50.00
  - Truck: ₹100.00
  - Motorcycle: ₹30.00

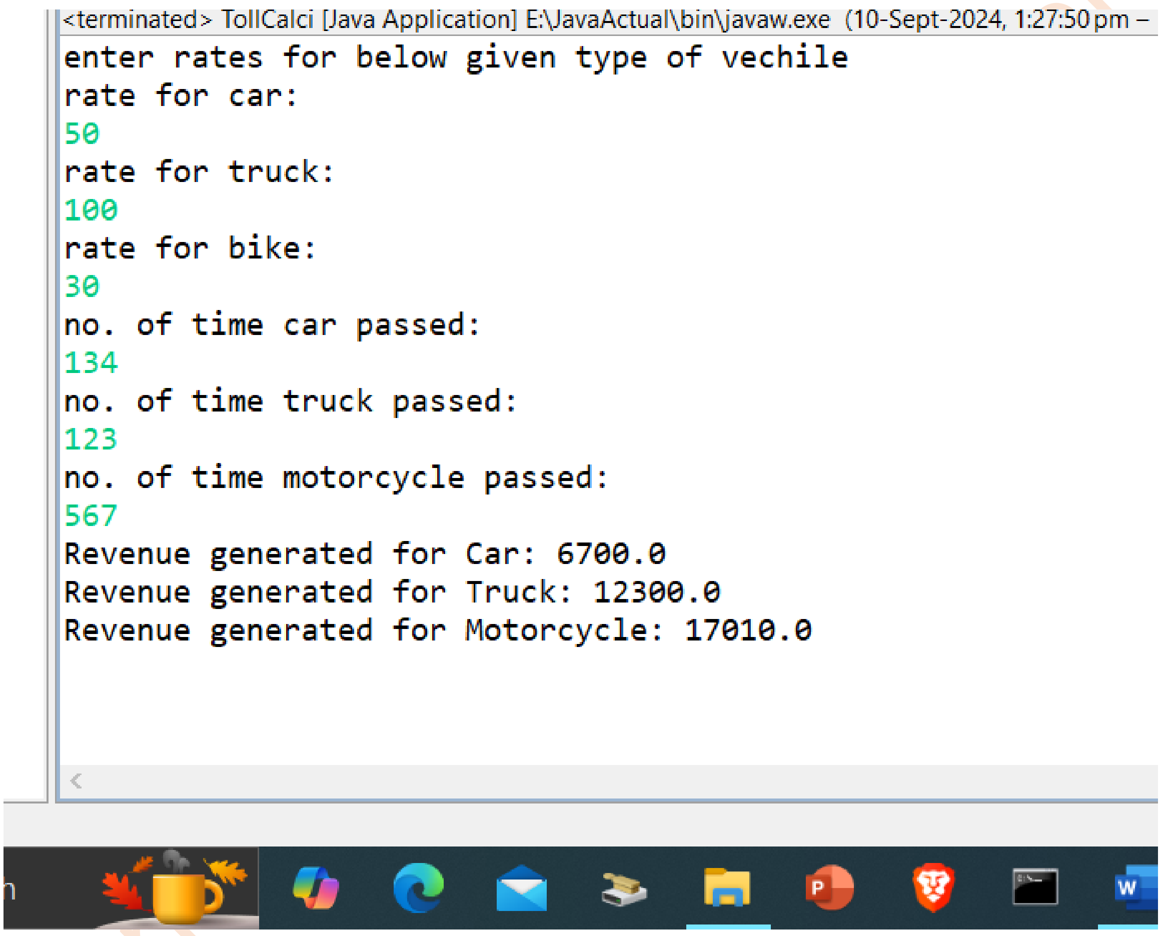
Define class TollBoothRevenueManager with methods acceptRecord, setTollRates, calculateRevenue & printRecord and test the functionality in main method.

Ans: **public class** TollCalci {

```
        float rc,rt,rm;
        void setTollRates() {
                Scanner sc =new Scanner(System.in);
```

```
		System.out.println("enter rates for below given type of vechile");
		System.out.println("rate for car: ");
		this.rc=sc.nextFloat();

		System.out.println("rate for truck: ");
		this.rt=sc.nextFloat();

		System.out.println("rate for bike: ");
		this.rm=sc.nextFloat();


	}

	void acceptRecord() {
		//2.	Accept the number of vehicles of each type passing through the toll
booth.
		Scanner sc =new Scanner(System.in);

		System.out.println("no. of time car passed: ");
		int pc=sc.nextInt();

		System.out.println("no. of time truck passed: ");
		int pt=sc.nextInt();

		System.out.println("no. of time motorcycle passed: ");
		int pm=sc.nextInt();


		calculateRevenue(pc,pt,pm);
		sc.close();
	}

	void calculateRevenue(int pc,int pt,int pm) {
		float revC=rc*pc;
		float revT=rt*pt;
		float revM=rm*pm;

		printRecord(revC,revT,revM);
	}
	public void printRecord(float revC, float revT, float revM) {
		// TODO Auto-generated method stub
		System.out.println("Revenue generated for Car: "+revC);
		System.out.println("Revenue generated for Truck: "+revT);
		System.out.println("Revenue generated for Motorcycle: "+revM);
	}
```

```
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                TollCalci t=new TollCalci();
                t.setTollRates();
                t.acceptRecord();

        }

}
```



```
<terminated> TollCalci [Java Application] E:\JavaActual\bin\javaw.exe  (10-Sept-2024, 1:27:50 pm –
enter rates for below given type of vechile
rate for car:
50
rate for truck:
100
rate for bike:
30
no. of time car passed:
134
no. of time truck passed:
123
no. of time motorcycle passed:
567
Revenue generated for Car: 6700.0
Revenue generated for Truck: 12300.0
Revenue generated for Motorcycle: 17010.0
```